SIMULATED CIRCUIT NODE INITIALIZING AND MONITORING

Technical Field of the Invention

The present invention relates generally to electronic circuitry simulation programs and in particular to program modules that control initial conditions and provide error detection for simulated circuit nodes.

Copyright Notice/Permission

Background of the Invention

Electronics have become part of almost every aspect of modern life, from television sets to toasters. As technology progresses, electronic circuits become increasingly complex. Designing a circuit with hundreds or even thousands of individual electronic components has become a great challenge for circuit designers. The challenge becomes even greater when many iterations of prototypes are required to test out the circuits and prove their viability. As the number of components multiply, building the actual circuit and modifying it becomes an expensive and time consuming proposition.

To overcome the problems associated with working with real components, circuit designers turned to computers to aid them with their designs. By using a computer, designers could create "virtual" circuits. This enabled designers to create, modify and test circuits without actually building the circuit with real components. Hardware Description Languages (HDL) were developed to aid designers in creating the virtual circuits. HDL programs allow designers to choose from a standardized set of instructions while designing their circuits on computers. Many different HDL programs have been written, each with varying capabilities. Among these HDL programs is Verilog HDL which is widely used and is considered an industry standard among circuit designers.

Verilog HDL uses the computer to simulate complex circuit designs with various inputs and outputs. It uses "modules" or lines of code representing a desired function as building blocks to create the simulation. The designer builds simulated circuitry by simply selecting and connecting the desired modules. Individual "nodes" or electrical connection points of the circuit can be analyzed while changing input variables to the circuit. The designer can even initialize nodes of the circuitry to various logic values before the simulation is started. Currently, however, this initial node parameter is not stabilized and will "float" or change to an unknown logic value during the simulation run. The designer has no way of knowing when or if the initial condition of the node has changed. If this is crucial to the design, it presents a serious, time consuming issue to surmount.

One method of overcoming this problem is to write a separate simulation program and use Verilog's Programming Language Interface (PLI) to export the desired node value to the separate simulation program. However, this method requires that a specific co-simulation program be written by a programmer for each Verilog simulation. This is both costly and time consuming and slows down the design cycle. Accordingly, what is needed is a simulation initialization and monitoring solution that is both quick and cost effective to use in circuit design, especially when using Verilog HDL.

Summary of the Invention

The present invention provides a method and apparatus for initializing and monitoring a simulated circuit node in a simulation system. An initial condition

(IC) behavior module is provided in a hardware definition language simulation system which operates in two phases. In the first phase, the IC module sets an initial logic condition onto a user-selected node which is to be monitored. The IC module will release the initial condition and then test the node value to determine if the simulation system is able to resolve the node. Alternatively, the IC module may release the node if a user-defined IC time period passes. In the second phase, the IC module monitors the node and reports an error detection message if the simulated node value becomes unacceptable.

These and other embodiments, aspects, advantages and features of the present invention will be set forth in part in the description which follows, and in part will become apparent to those skilled in the art by reference to the following description of the invention and referenced drawings or by practice of the invention. The aspects, advantages and features of the invention are realized and attained by means of the instrumentalities, procedures and combinations particularly pointed out in the appended claims.

Brief Description of the Drawings

Figure 1 is a diagram depicting an environment of one embodiment of the present invention.

Figure 2 is a block diagram illustrating how the present invention is selected for use in one embodiment.

Figure 3 is a block diagram of one embodiment of the present invention.

Figure 4 is a functional flow diagram of the embodiment shown in Fig. 3.

Figure 5 is a block diagram of another embodiment of the present invention.

Figure 6 is an expanded view of the inputs of the embodiment of Fig. 5.

Figure 7 is an expanded view of the outputs of the embodiment of Fig. 5.

Figure 8 is a functional flow diagram of the embodiment of Fig. 5.

Figure 9 is a depiction of examples of the media on which the present invention can reside.

Figure 10 is another embodiment of the present invention in a computer system.

Detailed Description of the Embodiments

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific preferred embodiments in which the inventions may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical or software changes may be made without departing from the scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims.

The present invention is used in conjunction with a computer software program and, thus, requires some understanding of how the computer program operates. In order to provide a clearer understanding, a description of how the computer software program is used relevant to the invention is provided, followed by detailed descriptions of the present invention's embodiments.

Verilog HDL is a hardware description language for designing and documenting digital logic systems. The language is used to create programs that simulate the interaction between components. The programs are composed of one or more independently developed modules that are not combined until the program is linked. A single module can contain one or more routines which are sections of a program that perform a particular task or tasks. Generally, it is the program user who decides which modules will be linked to perform an executable program. Modules are usually chosen from a list in a way that when linked they will perform a desired behavior. With programs such as Verilog HDL, the modules can be linked to form complex logic systems with complicated behavioral characteristics. The program user can define intricate input logic levels and create detailed electronic components that alter the logic levels to produce an output logic level with the user's designed specifications.

Often it is desirable to be able to monitor a logic level within the circuitry itself and not just at the final output. The user may also require that an initial logic level, also known as an initial condition, be set on certain nodes within the system to be simulated before executing the program. In order to accomplish these tasks, a separate simulation program is usually written using the Programming Language Interface (PLI) function of the Verilog HDL. The PLI consists of an interface mechanism, a set of routines to interact with the simulation environment and a set of routines to access the Verilog HDL internal data structures, providing dynamic interaction with the electronic circuitry simulation and the data structures. The disadvantages of PLI is that a separate co-simulation program is required and often times the interaction of the PLI program causes the Verilog HDL program to slow down to allow the data to be transferred back and forth between the two programs. This limits any "real time" analysis of the electronic circuitry.

The current invention is a behavior module that performs initializing and monitoring and manipulating logic level functions. Because it is a module, it is selectable from within a hardware design language program, such as Verilog HDL, and eliminates the need and constraints of a PLI co-simulation program. This initial condition (IC) module provides a means to initialize logic nodes (force an initial logic condition) to a desired logic level and to release the node if certain conditions are met. The release of the node means that the simulation program is free to change the logic value of the node if simulation warrants a change in the logic value. Before release, the logic value of the node is forced.

The IC module will release the node if certain conditions are met. One condition is the passage of time. The IC module may be programmed to hold the initial condition of the node for a user-defined period of time. After expiration of that user-defined time period, the node is released. Another release of the node by the IC module is when the simulation system is able to resolve the logic level of the node. Resolving the node means that the inputs to the node are known and therefore the logic value of the node can be determined through simulation by the simulation program. Thus, the IC module will release the node if the node can be resolved. Those skilled in the art will recognize other conditions upon which the node may be released.

The IC module can also monitor the node after it is released. If the node has been released, and the logic value of the node later becomes unknown, the IC module flags this as an error condition which may be the result of a logic design error. When monitoring the node, one skilled in the art will appreciate that an error condition from the IC module can be used to trigger error messages, halt simulation programs, trigger other parts of the simulated circuitry or viewed on a display device.

The convenience of having the ability to select a module to perform these functions also removes the timing constraints imposed by PLI co-simulation programs. The module allows for "real time" monitoring and analysis of the node voltage during the simulation program execution. The time and expense of writing a separate co-simulation program for the PLI is completely eliminated. A single user can insert the module where needed as they construct the simulated circuit, saving both time and money.

Figure 1 is an example of an environment in which the present invention operates. A computer system 10 contains a hardware definition language program 20 which is stored on the computer system 10 hard drive in this example. Within the HDL program, is an executable simulation program 30 and a resource set of available modules 40. A user selects which modules to use from the available modules 40 and these modules become part of the executable simulation program 30. The selected modules 50 then interact with simulation program instructions 60 to form the executable simulation program 30.

Figure 2 shows an enlarged list of selected modules 50 and an enlarged list of available modules 40. Normally, not all of the available modules 40 are executed as part of the executable simulation program 30. The present invention behavior module is selectable from the list of available modules 40. In the example of

Figure 2, the present invention module is called an initial condition module or an IC module 110. To use the IC module 110, the user selects it from the list of available modules 40 and then it becomes part of the selected modules 50 which are used during the execution of the simulation program 30. Referring back to Figure 1, when the simulation program 30 is executed, a simulated circuit 70 with the user's design characteristics is created. Generally, the simulated circuit 70 will have many connections or nodes such as node 80. This simulated circuit 70 is then tested by varying parameters such as the initial condition value of node 80.

In one embodiment shown in Figure 3, the IC module 110 contains routines to allow an initial node value input 150, a forced node value output 160, a simulated node value input 180 and an error indication output 190. The initial node value input 150 is either a user definable variable or a variable determined from within the executable simulation program or external to the executable simulation program. The initial node value input 150 is the initial logic value that the node is set to at the start of the simulation program. The initial logic condition may be a logic one, logic zero or high-impedance. The IC module 110 forces the logic level 160 for the node 80 if conditions warrant the forcing. If the forced initial condition is still warranted (as described above), the simulated circuit 170 is then simulated based upon the forced value 160 for that node.

The forced node value 160 is then released to allow the simulated node value 180 to be input to the IC module 110. The simulated node value is monitored by the IC module monitoring routine during the simulation program execution and after the release of the node. If the node value changes to an unknown value, the IC module reports an error detection via the error indication output 190. To one skilled in the art, it can be appreciated that the error detection can be triggered by other events besides a node value that is unknown. For example, the error condition can be triggered by the node value remaining unchanged for a length of time. The modules are linkable, and it is within the scope of the claimed invention to have several modules interconnected or one module linked to multiple nodes.

Figure 4 shows a data flow chart for the embodiment show in Figure 3. The operation of the IC module 110 can be viewed in two phases. In the first phase of operation, an initial node value (logic level) is set at 300 for a selected node. The logic level for the selected node value is held at that value until it is released by the IC module. As the simulation begins, the value is forced to the initial condition at 310 to hold the logic value on the node. As the simulation progresses to the next time step, the node is then released at 315 to allow the simulation program to attempt to resolve the node value. The value of the node is then tested at 316 to determine if the simulation software was successful in resolving the node value. If the node value is indeterminable or unknown at 316, the IC module will force the initial condition 310 once again via 305 of Figure 4. If the simulation software is successful in resolving the value of the node, then IC module enters a second phase of operation to monitor the node value. In the alternative, if a user-defined IC manipulation time period expires, the IC module will enter the second phase.

In the second phase of operation, IC module 110 can continue to monitor the node value 320 until the end of simulation or until it is halted by a user-defined time limit, or by some other mechanism known to those skilled in the art. The node value is continually tested at 330 to determine if the node value changes to an unknown, indeterminable or otherwise unacceptable logic state. If the value is valid, the IC module 110 continues to monitor the node value at 320 via 325. If the value is unacceptable, an indication of this condition is made at 340 via 335 and the simulation continues as control is passed via 325 to continue to monitor the node at 320 after the unacceptable condition is noted at 340.

Another embodiment is shown in Figures 5-7. The IC module 205, as shown in Figure 5, includes inputs 200, outputs 210, node settings 206 and node data 207 interfaces. The inputs 200 contain additional routines to allow an IC manipulation time period 220, a hold node constant command 230, an IC error detection period 235 and an IC simulation run time 236 as shown in Figure 6. Additional routines are also included in outputs 210 to allow an error indication 190, a real time node value output 240 and an end of simulation node value output 250 as shown in Figure 7.

The IC manipulation time period 220 is set to force the simulation node 80 to contain the desired information within the desired time interval of simulation program execution. The hold node constant command 230 is used to pass the node settings 206 to the simulation circuitry 170 such that the node value remains constant when the release conditions are met in IC module 205. The hold node constant command 230 allows a user, an internal link or an external link to instruct the module to force the node value to a constant value for the duration of the simulation program execution or until a release condition such as at a user-defined IC manipulation period 220 or if resolved.

To one skilled in the art, it can be appreciated that the IC manipulation time input 220 can be used to control the length of time that the node is held to a constant value. The real time node value output 240 is the node value as monitored via the node value input 207 and is output for the duration of the error detection period 235. The IC error detection period 235 is used by the module to monitor the node logic value during the simulation program execution. To one skilled in the art, it can be appreciated that this input value can be used to limit the duration of the monitoring activities of the module even before the simulation program has fully executed.

The IC simulation run time input 236 is used by the module to determine the start and finish of the IC simulation program execution. This time is used to determine the end of simulation node value output 250 based on a user-defined period of time executing the simulation program.

The module outputs 210 are displayable by any number of means including CRT displays, printouts and any device used to convey information. The module outputs are also usable as inputs to other modules to provide further control of the simulation program or to trigger other events, internal or external to the simulation program. The module inputs 200 are input by any number of means including keyboards, touch screens and any device used to input information. The module inputs are also linkable to outputs of other modules to provide further control of the simulation program or to trigger other events.

Figure 8 shows a flow chart for the embodiment shown in Figures 5-7. In this embodiment, two general phases are shown. In the first phase, an initial node value, an IC manipulation time, a hold node constant command, an error detection period and an IC simulation run time are obtained 400 and then a simulated circuit node value for the node to be controlled/monitored is forced to equal to the initial node value 410. As the simulation progresses to the next time step, the IC module 205 then releases the node value at 415 to allow the simulation software of the simulated circuit 170 to attempt to resolve the value of the node. A check is then performed at 420 to determine if the node value has been resolved and is therefore valid. If the simulation software has been unable to resolve the node value to an acceptable or valid logic condition (e.g.: logic one, zero or high-impedance), the IC module 205 will once again force the initial condition on the node at 410. The simulation will then continue by once again releasing the node at 415 and testing again at 420. This loop will continue until the node resolves to a valid value. In the alternative, if a user-defined IC manipulation time period 220 passes, control will pass to the second phase to monitor the node value at 440. Those skilled in the art will readily recognize that what is a valid condition and what is invalid is determined by the needs of the user using the simulation program. In some applications, an invalid or unknown logic state may be acceptable.

If the node resolves to a valid logic state or if the user-defined IC manipulation time period 220 has passed, then phase two of the operation of the IC module 205 begins. The second phase shown in Figure 8 is the monitoring phase which begins at 440. At this point, the simulated circuit node is monitored 440 to catch any logic state change. If there is any change of the node value detected, the node value is tested to see if it is acceptable at 450. If the node value is acceptable (decision branch 455), the value of the node is output to the user at 460. If the node value is unacceptable (decision branch 465), an error indication is output 470. Afterwards, a check is performed at 480 to see if the error detection time period 235 is ended. If the IC error detection period 235 has not expired (decision branch 486), the control flow returns to monitoring the node at 440 and the cycle is repeated. If the user-defined error detection time period 235 has ended (decision branch 485), simulation advances to decision block 490 via path 485. If the IC simulation run time 236 is ended in 490, the final value of the node is output via 491 to the user at 499 and the IC simulation is ended at 500, otherwise the real time value can be output via 492 to the user at 495 and the check loop is repeated via 496 until the end of IC simulation run time 236 is complete. Those skilled in the art will readily recognize that what is a valid/acceptable condition and what is invalid/unacceptable condition is determined by the needs of the user using the simulation program. In some applications, an unknown logic state may be recognized as acceptable.

In another embodiment of the invention shown in Figure 9, the module is part of a list of instructions found on computer readable media such as diskettes, hard drives or random access memory. In yet another embodiment of the invention shown in Figure 10, the module is part of a computer system used to design simulated electronic circuitry.

In summary, the present invention is a module that is easily selected from within the hardware definition language program. It has inputs for setting various parameters used to control and monitor circuit nodes from within a simulation program. The module's outputs provide both control of the circuit node voltage and also information that is displayable or is used to provide further control of other modules.

An advantage of the present invention is that the module is part of the simulation program and not a separate co-simulation program. This allows faster simulation and eliminates the need to write another program. Both of these aspects save time and money. Another advantage is the ease of use afforded the user due to the fact that the module is selectable as part of the simulation program and can be incorporated "on the fly" as the circuit design is being developed. This ensures greater accuracy during troubleshooting and also ensures that the designer is the one who chooses the correct monitoring points.

Other incorporations and uses of the module will be apparent to those skilled in the art. It is to be understood that the above description is intended to be illustrative and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.